# Assignment#2 딥러닝 보고서

20181592 김재민

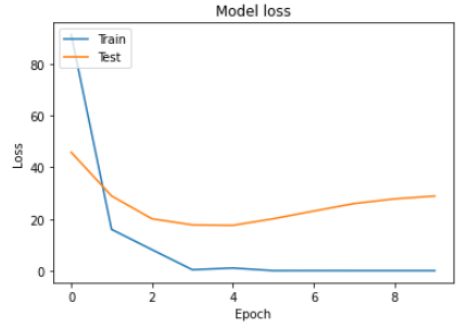
## 1. overfitting 현상을 train loss와 test loss를 출력해서 확인하고 해결하기

Overfitting 즉, 과적합이란 모델이 train data에 지나치게 적응되어 그 외의 새로운 데이터에는 제대로 대응하지 못하는 상태를 말한다.

과적합이 일어나는 경우는 다양하지만 주로 일어나는 이유 중 하나인 train data가 적은 경우를 충족시켜 보았다. 일부러 MNIST 데이터셋에서 6만 개 중 200개만을 train 데이터로 사용했다.

과적합의 발생 여부는 train loss와 test loss를 출력해서 확인해본다.

```
Epoch 1/10
5/5 [==============================] - 1s 53ms/step - loss: 91.4148 - accuracy: 0.3133 - val_loss: 45.8248 - val_accuracy: 0.5400
Epoch 2/10
5/5 [==============================] - 0s 12ms/step - loss: 15.9773 - accuracy: 0.8067 - val_loss: 28.9110 - val_accuracy: 0.6600
Epoch 3/10
5/5 [==============================] - 0s 18ms/step - loss: 8.1114 - accuracy: 0.9067 - val_loss: 20.0735 - val_accuracy: 0.7600
Epoch 4/10
5/5 [==============================] - 0s 12ms/step - loss: 0.3536 - accuracy: 0.9867 - val_loss: 17.7141 - val_accuracy: 0.8400
Epoch 5/10
5/5 [==============================] - 0s 12ms/step - loss: 1.0162 - accuracy: 0.9800 - val_loss: 17.5362 - val_accuracy: 0.8200
Epoch 6/10
5/5 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 20.0802 - val_accuracy: 0.8000
Epoch 7/10
5/5 [==============================] - 0s 12ms/step - loss: 0.0094 - accuracy: 0.9933 - val_loss: 23.0549 - val_accuracy: 0.8000
Epoch 8/10
5/5 [==============================] - 0s 13ms/step - loss: 5.1614e-04 - accuracy: 1.0000 - val_loss: 25.9606 - val_accuracy: 0.7800
Epoch 9/10
5/5 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 27.7670 - val_accuracy: 0.7800
Epoch 10/10
5/5 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 28.8851 - val_accuracy: 0.7800
```




Train loss는 학습을 하면서 거의 0에 수렴하며 낮아졌지만, test loss는 감소하다가 어느 순간부터 점차 증가하는 것을 확인할 수 있고 또한 각각의 loss가 차이가 꽤 나는 것을 확인할 수 있다. 이를 통해 train에 비해 test 성능이 낮고 그 차이가 큰 즉, overfitting이 발생했다는 것을 알 수 있다.

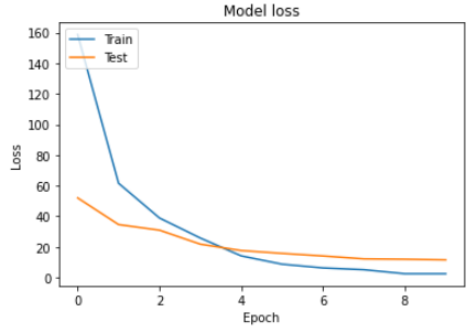
이러한 overfitting을 해결하기 위해 드롭아웃(dropout)이라는 기법을 사용했다.

Dropout은 train 과정 중에 일부 뉴런을 삭제하고 일부 뉴런은 sub group을 맺어서 학습하는 기법으로, 이와 같이 학습하게 되면 feature가 적게 반영되어 variance를 낮출 수 있고 이러한 적은 variance로 예측을 서로 도와주며 overfitting을 막아준다. (앙상블 방법과 비슷하다.)

`keras.layers.Dropout(0.5),` 부분을 통해 0.5라는 값을 주어서 뉴런의 절반만 학습에 이용하도록 했다.

Dropout 방식을 적용하여 train loss와 test loss를 다시 출력해본다.

```
Epoch 1/10
5/5 [==============================] - 1s 45ms/step - loss: 158.6729 - accuracy: 0.1933 - val_loss: 52.0264 - val_accuracy: 0.4000
Epoch 2/10
5/5 [==============================] - 0s 10ms/step - loss: 61.5976 - accuracy: 0.3200 - val_loss: 34.6264 - val_accuracy: 0.5400
Epoch 3/10
5/5 [==============================] - 0s 9ms/step - loss: 38.9192 - accuracy: 0.5000 - val_loss: 30.9659 - val_accuracy: 0.5800
Epoch 4/10
5/5 [==============================] - 0s 11ms/step - loss: 25.8300 - accuracy: 0.6200 - val_loss: 21.8224 - val_accuracy: 0.6400
Epoch 5/10
5/5 [==============================] - 0s 11ms/step - loss: 14.1709 - accuracy: 0.7200 - val_loss: 17.6583 - val_accuracy: 0.6800
Epoch 6/10
5/5 [==============================] - 0s 15ms/step - loss: 8.7538 - accuracy: 0.7867 - val_loss: 15.8004 - val_accuracy: 0.7200
Epoch 7/10
5/5 [==============================] - 0s 9ms/step - loss: 6.3131 - accuracy: 0.7933 - val_loss: 14.1155 - val_accuracy: 0.7000
Epoch 8/10
5/5 [==============================] - 0s 9ms/step - loss: 5.1596 - accuracy: 0.8267 - val_loss: 12.2256 - val_accuracy: 0.7400
Epoch 9/10
5/5 [==============================] - 0s 9ms/step - loss: 2.5345 - accuracy: 0.8733 - val_loss: 11.9791 - val_accuracy: 0.7400
Epoch 10/10
5/5 [==============================] - 0s 10ms/step - loss: 2.5333 - accuracy: 0.8800 - val_loss: 11.5698 - val_accuracy: 0.7400
```




Test loss가 이전과 다르게 우 하향을 보이고 있으며 train loss와의 격차도 이전과 다르게 크게 줄었음을 확인할 수 있다. 이를 통해 dropout 기법을 적용하여 overfitting을 억제하는 것에 성공했다고 할 수 있다.

## 2. CIFAR-10 dataset 모델 성능 비교

기존의 Baseline1 모델과 Baseline2 모델을 2가지 조건에서 학습해보았다.

Learning rate = 0.01 / Epochs = 10

```
Epoch: 0 | Loss: 2.1718 | Train Accuracy: 28.24
Epoch: 1 | Loss: 2.1260 | Train Accuracy: 32.89
Epoch: 2 | Loss: 2.1130 | Train Accuracy: 34.06
Epoch: 3 | Loss: 2.1008 | Train Accuracy: 35.44
Epoch: 4 | Loss: 2.0948 | Train Accuracy: 35.91
Epoch: 5 | Loss: 2.0865 | Train Accuracy: 36.93
Epoch: 6 | Loss: 2.0823 | Train Accuracy: 37.17
Epoch: 7 | Loss: 2.0760 | Train Accuracy: 37.79
Epoch: 8 | Loss: 2.0735 | Train Accuracy: 38.03
Epoch: 9 | Loss: 2.0718 | Train Accuracy: 38.27
Test Accuracy: 39.60
```

<Baseline1>

```
Epoch: 0 | Loss: 1.9335 | Train Accuracy: 27.76
Epoch: 1 | Loss: 1.5750 | Train Accuracy: 41.85
Epoch: 2 | Loss: 1.4685 | Train Accuracy: 46.37
Epoch: 3 | Loss: 1.3842 | Train Accuracy: 49.41
Epoch: 4 | Loss: 1.3276 | Train Accuracy: 52.29
Epoch: 5 | Loss: 1.2939 | Train Accuracy: 53.54
Epoch: 6 | Loss: 1.2524 | Train Accuracy: 54.87
Epoch: 7 | Loss: 1.2285 | Train Accuracy: 56.16
Epoch: 8 | Loss: 1.2003 | Train Accuracy: 57.15
Epoch: 9 | Loss: 1.1742 | Train Accuracy: 58.35
Test Accuracy: 60.91
```

<Baseline2>

Learning rate = 0.01 / Epochs = 20

```
Epoch: 0 | Loss: 2.1698 | Train Accuracy: 28.33
Epoch: 1 | Loss: 2.1161 | Train Accuracy: 34.10
Epoch: 2 | Loss: 2.0962 | Train Accuracy: 35.96
Epoch: 3 | Loss: 2.0855 | Train Accuracy: 36.89
Epoch: 4 | Loss: 2.0780 | Train Accuracy: 37.80
Epoch: 5 | Loss: 2.0706 | Train Accuracy: 38.55
Epoch: 6 | Loss: 2.0675 | Train Accuracy: 38.80
Epoch: 7 | Loss: 2.0634 | Train Accuracy: 39.09
Epoch: 8 | Loss: 2.0638 | Train Accuracy: 38.93
Epoch: 9 | Loss: 2.0610 | Train Accuracy: 39.27
Epoch: 10 | Loss: 2.0576 | Train Accuracy: 39.78
Epoch: 11 | Loss: 2.0576 | Train Accuracy: 39.69
Epoch: 12 | Loss: 2.0554 | Train Accuracy: 39.91
Epoch: 13 | Loss: 2.0552 | Train Accuracy: 39.89
Epoch: 14 | Loss: 2.0529 | Train Accuracy: 40.16
Epoch: 15 | Loss: 2.0509 | Train Accuracy: 40.50
Epoch: 16 | Loss: 2.0521 | Train Accuracy: 40.29
Epoch: 17 | Loss: 2.0506 | Train Accuracy: 40.41
Epoch: 18 | Loss: 2.0499 | Train Accuracy: 40.56
Epoch: 19 | Loss: 2.0504 | Train Accuracy: 40.46
Test Accuracy: 41.94
```

<Baseline1>

```
Epoch: 0 | Loss: 1.9207 | Train Accuracy: 28.43
Epoch: 1 | Loss: 1.5767 | Train Accuracy: 41.86
Epoch: 2 | Loss: 1.4504 | Train Accuracy: 47.34
Epoch: 3 | Loss: 1.3633 | Train Accuracy: 51.07
Epoch: 4 | Loss: 1.3082 | Train Accuracy: 52.95
Epoch: 5 | Loss: 1.2681 | Train Accuracy: 54.62
Epoch: 6 | Loss: 1.2241 | Train Accuracy: 56.40
Epoch: 7 | Loss: 1.2085 | Train Accuracy: 56.95
Epoch: 8 | Loss: 1.1780 | Train Accuracy: 58.12
Epoch: 9 | Loss: 1.1546 | Train Accuracy: 58.98
Epoch: 10 | Loss: 1.1348 | Train Accuracy: 59.95
Epoch: 11 | Loss: 1.1102 | Train Accuracy: 60.70
Epoch: 12 | Loss: 1.0931 | Train Accuracy: 61.38
Epoch: 13 | Loss: 1.0917 | Train Accuracy: 61.21
Epoch: 14 | Loss: 1.0669 | Train Accuracy: 62.50
Epoch: 15 | Loss: 1.0654 | Train Accuracy: 62.44
Epoch: 16 | Loss: 1.0460 | Train Accuracy: 63.11
Epoch: 17 | Loss: 1.0455 | Train Accuracy: 63.16
Epoch: 18 | Loss: 1.0324 | Train Accuracy: 63.42
Epoch: 19 | Loss: 1.0326 | Train Accuracy: 63.77
Test Accuracy: 66.27
```

<Baseline2>

Baseline1은 Baseline에 비해 정확도가 크게 뒤쳐졌다. Epochs를 증가시켰을 때는 정확도 차이가 더 벌어졌다.

NewModel1은 Baseline2에서 model의 capacity를 늘리기 위해 parameter를 늘려 학습을 진행했다.

NewModel2는 앞선 model들보다 conv층을 더 쌓는 방법을 통해 학습 parameter 수를 늘렸고, channel 수 역시 층마다 증가시켜주면서 parameter 수를 늘렸다. 그리고 Max Pooling을 NewModel2에 비해 추가하고 dropout을 사용하여 overfitting을 방지하고자 했다.

Learning rate = 0.01 / Epochs = 10

```
Epoch: 0 | Loss: 1.8243 | Train Accuracy: 32.66
Epoch: 1 | Loss: 1.4344 | Train Accuracy: 47.52
Epoch: 2 | Loss: 1.2631 | Train Accuracy: 54.22
Epoch: 3 | Loss: 1.1413 | Train Accuracy: 59.18
Epoch: 4 | Loss: 1.0620 | Train Accuracy: 61.90
Epoch: 5 | Loss: 1.0097 | Train Accuracy: 64.04
Epoch: 6 | Loss: 0.9596 | Train Accuracy: 65.94
Epoch: 7 | Loss: 0.9204 | Train Accuracy: 67.68
Epoch: 8 | Loss: 0.9000 | Train Accuracy: 68.27
Epoch: 9 | Loss: 0.8710 | Train Accuracy: 69.35
Test Accuracy: 71.50
```

<NewModel1>

```
Epoch: 0 | Loss: 1.9173 | Train Accuracy: 26.98
Epoch: 1 | Loss: 1.4350 | Train Accuracy: 46.89
Epoch: 2 | Loss: 1.1505 | Train Accuracy: 58.67
Epoch: 3 | Loss: 0.9826 | Train Accuracy: 65.36
Epoch: 4 | Loss: 0.8699 | Train Accuracy: 69.68
Epoch: 5 | Loss: 0.7784 | Train Accuracy: 72.98
Epoch: 6 | Loss: 0.7218 | Train Accuracy: 75.33
Epoch: 7 | Loss: 0.6728 | Train Accuracy: 76.77
Epoch: 8 | Loss: 0.6245 | Train Accuracy: 78.42
Epoch: 9 | Loss: 0.6040 | Train Accuracy: 79.29
Test Accuracy: 79.67
```

<NewModel2>

Learning rate = 0.01 / Epochs = 20

```
Epoch: 0 | Loss: 1.8080 | Train Accuracy: 33.15
Epoch: 1 | Loss: 1.4534 | Train Accuracy: 46.76
Epoch: 2 | Loss: 1.2965 | Train Accuracy: 53.25
Epoch: 3 | Loss: 1.1810 | Train Accuracy: 57.86
Epoch: 4 | Loss: 1.0998 | Train Accuracy: 60.81
Epoch: 5 | Loss: 1.0356 | Train Accuracy: 63.35
Epoch: 6 | Loss: 0.9825 | Train Accuracy: 65.31
Epoch: 7 | Loss: 0.9525 | Train Accuracy: 66.36
Epoch: 8 | Loss: 0.9212 | Train Accuracy: 67.53
Epoch: 9 | Loss: 0.8942 | Train Accuracy: 68.25
Epoch: 10 | Loss: 0.8721 | Train Accuracy: 69.26
Epoch: 11 | Loss: 0.8537 | Train Accuracy: 69.80
Epoch: 12 | Loss: 0.8362 | Train Accuracy: 70.67
Epoch: 13 | Loss: 0.8217 | Train Accuracy: 71.12
Epoch: 14 | Loss: 0.8058 | Train Accuracy: 71.82
Epoch: 15 | Loss: 0.7885 | Train Accuracy: 72.50
Epoch: 16 | Loss: 0.7763 | Train Accuracy: 72.68
Epoch: 17 | Loss: 0.7660 | Train Accuracy: 72.96
Epoch: 18 | Loss: 0.7556 | Train Accuracy: 73.52
Epoch: 19 | Loss: 0.7551 | Train Accuracy: 73.63
Test Accuracy: 73.92
```

<NewModel1>

```
Epoch: 0 | Loss: 1.9226 | Train Accuracy: 26.77
Epoch: 1 | Loss: 1.4262 | Train Accuracy: 47.34
Epoch: 2 | Loss: 1.1676 | Train Accuracy: 58.17
Epoch: 3 | Loss: 0.9887 | Train Accuracy: 65.31
Epoch: 4 | Loss: 0.8700 | Train Accuracy: 70.11
Epoch: 5 | Loss: 0.7856 | Train Accuracy: 72.81
Epoch: 6 | Loss: 0.7278 | Train Accuracy: 75.20
Epoch: 7 | Loss: 0.6735 | Train Accuracy: 76.66
Epoch: 8 | Loss: 0.6304 | Train Accuracy: 78.37
Epoch: 9 | Loss: 0.5916 | Train Accuracy: 79.61
Epoch: 10 | Loss: 0.5613 | Train Accuracy: 80.89
Epoch: 11 | Loss: 0.5363 | Train Accuracy: 81.58
Epoch: 12 | Loss: 0.5137 | Train Accuracy: 82.27
Epoch: 13 | Loss: 0.4986 | Train Accuracy: 82.85
Epoch: 14 | Loss: 0.4701 | Train Accuracy: 84.02
Epoch: 15 | Loss: 0.4573 | Train Accuracy: 84.26
Epoch: 16 | Loss: 0.4332 | Train Accuracy: 85.13
Epoch: 17 | Loss: 0.4231 | Train Accuracy: 85.39
Epoch: 18 | Loss: 0.4085 | Train Accuracy: 85.93
Epoch: 19 | Loss: 0.4007 | Train Accuracy: 86.08
Test Accuracy: 84.61
```

<NewModel2>

모든 model을 비교해보았다.

[Learning rate = 0.01 / Epochs = 10]

| | Train Loss | Train Accuracy | Test Accuracy |
|---|---|---|---|
| Baseline1 | 2.07 | 38.27 | 39.60 |
| Baseline2 | 1.17 | 58.35 | 60.91 |
| NewModel1 | 0.87 | 69.35 | 71.50 |
| NewModel2 | 0.60 | 79.29 | 79.67 |

[Learning rate = 0.01 / Epochs = 20]

| | Train Loss | Train Accuracy | Test Accuracy |
|---|---|---|---|
| Baseline1 | 2.05 | 40.46 | 41.94 |
| Baseline2 | 1.03 | 63.77 | 66.27 |
| NewModel1 | 0.76 | 73.63 | 73.92 |
| NewModel2 | 0.40 | 86.08 | 84.61 |

Baseline1의 성능이 가장 떨어졌고 그 다음은 Baseline2였다. 이러한 Baseline2의 parameter를 늘려 만든 NewModel1은 Baseline2에 비해 성능이 증가한 모습을 보였다.

마지막으로 앞선 모든 model들에 비해 conv층 증가와 channel 수 증가를 통해 parameter 수를 늘려주고, Max pooling 추가와 dropout을 사용해 overfitting 방지 노력까지한 NewModel2는 가장 뛰어난 성능을 보여줬다.

또한 Epochs의 증가를 통해 모든 model들의 성능을 조금씩 향상시킬 수 있음을 확인했다.